Especificación Formal de un Modelo de Detección de Intrusos basado en la Teoría Sistemas Multi-Agentes

Iren Lorenzo Fonseca[1], Francisco Maciá Pérez[2], Francisco José Mora Gimeno[2], Diego Marcos Jorquera[2], Juan Antonio Gil Martínez[2], Rogelio Lau Fernández[1]

[1]Centro de Estudio de Ingeniería y Sistema
Instituto Superior Politécnico José Antonio Echevarría
{ilorenzo, lau}@ceis.cujae.edu.cu

[2] Departamento de Tecnología Informática y Computación.
Universidad de Alicante.
AP.99-03080, Alicante, España
{pmacia, jmora, dmarcos, gil}@dtic.ua.es
http://www.dtic.ua.es

Resumen. Los Sistemas de Detección de Intrusos (IDS) han demostrado ser una herramienta de gran valor para hacerle frente a los problemas actuales de seguridad informática. Los Sistemas de Detección de Intrusos de Red (IDS) han comenzado a ser una herramienta de gran valor para la seguridad informática. Estos IDS deben contar, desde la etapa de modelado, con características que hagan viable su implantación en las condiciones actuales destacándose eficiencia, capacidad de detección, dinamismo, escalabilidad y mantenimiento mínimo. La teoría de Sistemas Multi-Agentes (MAS) brinda un Marco formal adecuado para el modela de este tipo de sistemas. Por esta razón, el presente artículo describe formalmente, a través de un MAS, un Modelo IDS que utiliza Redes Neuronales Artificiales y el Análisis de Componentes Principales con el objetivo de lograr eficiencia y espectro de clasificación.

1 Introducción

La sociedad actual ha alineado su desarrollo con el desarrollo de la TI. Es precisamente el uso de las tecnologías de la información el que condiciona el buen funcionamiento de la gran mayoría de las ramas sociales [1-3]. Muestra de ello es el incremento del uso de estas tecnologías a nivel empresarial como se muestra en el gráfico de la figura 1.Dentro de este contexto, mantener los niveles de seguridad informática ha pasado a ser una tarea de primer orden.

Este hecho ha potenciado el desarrollo de investigaciones centradas en la obtención de herramientas de seguridad que se ajusten a los requerimientos actuales. Dentro de estas, los Sistemas de Detección de Intrusos (IDS), han destacado por su enfoque de brindar seguridad en el último nivel [4-6]. No obstante, este tipo de herramientas precisa de un conjunto de rasgos que hagan factible su implantación, estando la capacidad de detección, la eficiencia, la escalabilidad, la adaptabilidad dinámica y la autogestión entre las más perseguidas. Es importante, que todos estos requisitos sean concebidos a nivel de modelo, de manera que el IDS sea desarrollado partiendo de

ellos desde su concepción primaria. Por tanto, el modelado de un IDS sobre bases formales garantiza un diseño correcto del mismo, repercutiendo esto de manera directa sobre el producto final.

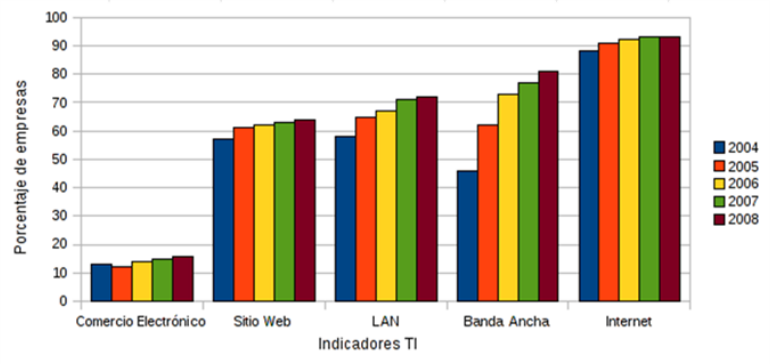

Fig. 1. Evolución del uso de las TI en las empresas [7]**.**

Teniendo en cuenta que los principales requerimientos perseguidos, en este artículo se modela un IDS desde el punto de vista funcional, con un enfoque distribuido y centrado en los responsables de llevar a cabo las funcionalidades. Resumiendo todos los factores descritos podemos concretar las principales características del Modelo de IDS (MIDS) como:

- Se trata de un en entorno distribuido.
- Se busca un modelo centrado en los actores responsables de la ejecución de los procesos.
- Se pretende un sistema escalable y dinámico.

La teoría de agentes y más concretamente los sistemas multiagente proporcionan un marco formal básico idóneo para los requerimientos descritos. Con una capacidad expresiva muy notables para abordar escenarios tan complejos y totalmente dirigidos por los agentes. En términos prácticos, el resultado final del *MIDS* es un algoritmo distribuido de detección definido a través de un Sistema Multi-Agente (MAS) que pueda ser implementado y ejecutado por un sistema informático de forma eficaz. La vista general del modelo se muestra en la figura 2.

Este modelo global de agentes contempla a la red de computadoras como un sistema de acción y reacción sobre el cual actúan los agentes (A) desarrollando tareas (T) con la intención (Γ) de llevar al sistema desde su estado actual ($\in$) hacia un estado considerado seguro ($_{seg} \in _{seg}$).

De esta manera, el presente artículo describe un modelo IDS que utiliza Análisis de Componentes Principales (PCA) como algoritmo de reducción de características (ref Reduc. Carac) y un Mapa Auto-Organizado (SOM) para la clasificación (ref ANN). El modelado de este *MIDS* se lleva a cabo a través de las formalizaciones expresadas en el marco formal de [8]. En este caso, el *MIDS* es el mundo que se quiere describir, siendo el *Modelo de red*, el entorno que contiene el mundo. La red será regulada por diferentes agentes, los Agentes Reguladores, cuyo objetivo común es detectar ataques

de manera eficiente. De manera que el MIDS se puede expresar formalmente como en la ecuación 1.

$$MIDS = \langle ModRed, A \rangle \quad (1)$$

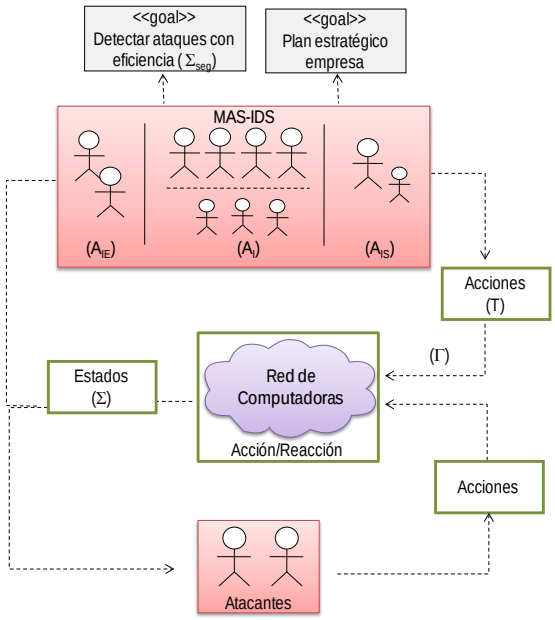

Fig. 2. Vista general de MIDS desde el punto de vista del Modelo funcional.

De esta forma a lo largo del artículo se profundizará sobre cada uno de los elementos que se proponen para la formalización del modelo

2 Modelo de Red

El Modelo de Red (*ModRed*) se describe como el conjunto de estados del mundo, tareas e influencias.

$$ModRed = \langle ,T, \rangle \quad (2)$$

Los estados del mundo describen los principales conceptos que se manejan en el entorno a regular y las tareas son las acciones que ejecutan los agentes. Por la importancia de ambos, éstos serán descritos con mayor nivel de detalles en las secciones siguientes. No se le dedica una sección a las influencias ya que éstas no son más que la conjunción de las dos anteriores. Una influencia es la ejecución de una tarea sobre un estado del mundo, por tanto su existencia es de suma importancia para

entornos regulados por múltiples agentes, pero queda perfectamente definida a través de las tareas y los estados del mundo.

2.1 Estados del Mundo

El conjunto de estados del mundo contiene todos los elementos del entorno, siendo el encargado de definir los principales conceptos de la red, de forma que las agentes puedan percibirlos e influir sobre ellos.

Para representar los estados del mundo, se utiliza una representación ontológica. Debido a lo compleja que es la tarea de describir una red de computadoras a través de una ontología [9-13] para este caso, nos centramos solamente en las descripciones de los conceptos que utilizarán los agentes. Nuestro IDS toma como fuente de información el tráfico de red y más concretamente los paquetes que circulan por la misma. De esta forma, hemos recogido en una ontología que describe los estados del mundo, las principales conceptualizaciones y restricciones que existen entre los paquetes de red (IP, ICMP, UDP, TCP, tramas Ethernet, datos de aplicación) y sus cabeceras. Con este objetivo, la ontología se centra en dos conceptos fundamentales: cabeceras y paquetes, donde cada uno genera una jerarquía de relaciones.

La figura 3 muestra las relaciones jerárquicas establecidas, entre las cabeceras. En ésta, se parte del concepto más abstracto en primer nivel, para luego tener un segundo nivel que caracteriza a las cabeceras según la capa de comunicación hasta llegar a un último nivel que expresa los conceptos específicos de las cabeceras de los protocolos de interés para esta aplicación.

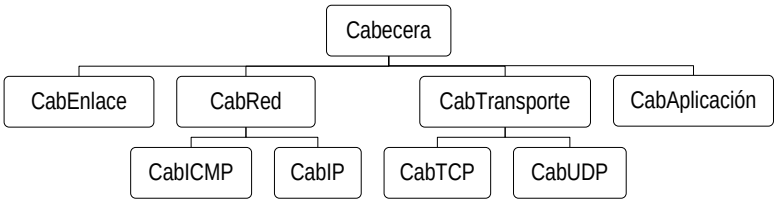

Fig. 3. Jerarquía de conceptos de cabecera.

Por otra parte, y de forma análoga, se expresa la jerarquía de conceptos de paquetes como se muestra en la figura 4.

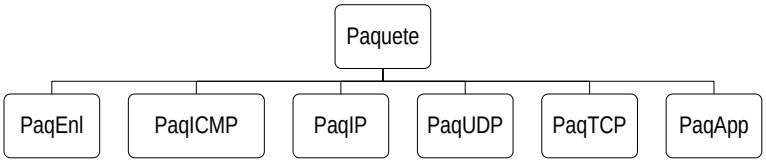

Fig. 4 Jerarquía de conceptos de paquete.

Adicionalmente se define una clase tráfico que controla, como atributos, las características del tráfico de red en general.

Utilizando la herramienta de edición de ontologías Protégé [14] las relaciones de herencias entre clases pueden observarse como se muestra en la figura. 5.

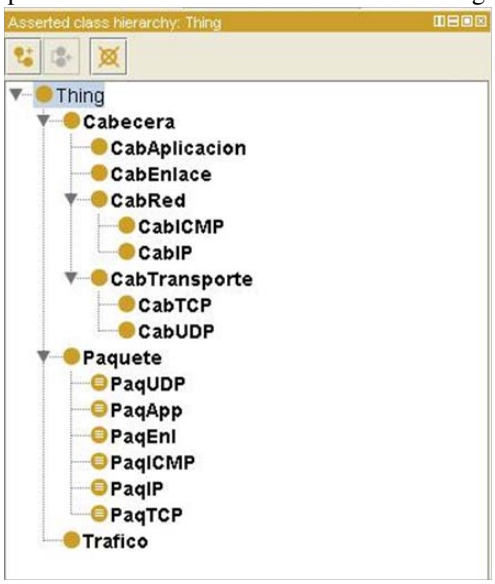

Fig. 5. Clases de la Ontología de Paquetes de Red.

Para relacionar los paquetes con las cabeceras se establecen dos relaciones principales con características inversas: *hasHeader* y *isHeaderOf* . Con ellas se definen las restricciones principales de cada clase y se fijan para *hasHeader* la clase *Paquete* como dominio y *Cabecera* como rango.

Una vez definidos los principales conceptos y relaciones, fue necesario precisar las restricciones que otorgan validez a cada clase. Por ello se acotan las relaciones entre cabeceras y paquetes y se añaden cardinalidades que permitieron dar mayor robustez y consistencia al modelado de los estados.

Un ejemplo de ello se muestra en la figura 6, donde se exponen las restricciones aplicadas a la clase *PaqTCP*. En éstas se especifica que un paquete TCP contiene un elemento de la cabecera TCP, otro de la cabecera IP y otro correspondiente a la información de la capa de enlace. Además se definen restricciones de cardinalidad con las cuales se garantiza la existencia de una única instancia de estos elementos dentro de la clase. Específicamente, la restricción de cardinalidad con la cabecera de transporte, evita que pueda existir una cabecera UDP dentro de un paquete TCP.

Fig. 6. Restricciones de la Clase PaqTCP.

El resto de las clases de paquetes se describen a través de restricciones similares y se transforman en clases definidas. De esta manera se logra que las restricciones sean de tipo necesaria y suficiente, para que el razonador pueda inferir nuevas relaciones de pertenencia a partir de las restricciones definidas. Las restricciones mostradas en la figura 6 pueden observarse en formato OWL [15] a continuación:

```
<EquivalentClasses>
        <Class URI="&PaqRed;PaqTCP"/>
        <ObjectIntersectionOf>
            <Class URI="&PaqRed;Paquete"/>
            <ObjectSomeValuesFrom>
                <ObjectProperty URI="&PaqRed;hasHeader"/>
                <Class URI="&PaqRed;CabEnlace"/>
            </ObjectSomeValuesFrom>
            <ObjectSomeValuesFrom>
              <ObjectProperty URI="&PaqRed;hasHeader"/>
                  <Class URI="&PaqRed;CabIP"/>
            </ObjectSomeValuesFrom>
            <ObjectSomeValuesFrom>
               <ObjectProperty URI="&PaqRed;hasHeader"/>
                <Class URI="&PaqRed;CabTCP"/>
            </ObjectSomeValuesFrom>
            <ObjectExactCardinality cardinality="1">
               <ObjectProperty URI="&PaqRed;hasHeader"/>
                <Class URI="&PaqRed;CabEnlace"/>
            </ObjectExactCardinality>
            <ObjectExactCardinality cardinality="1">
            <ObjectProperty URI="&PaqRed;hasHeader"/>
                <Class URI="&PaqRed;CabIP"/>
            </ObjectExactCardinality>
            <ObjectExactCardinality cardinality="1">
                <ObjectProperty URI="&PaqRed;hasHeader"/>
               <Class URI="&PaqRed;CabTransporte"/>
            </ObjectExactCardinality>
        </ObjectIntersectionOf>
</EquivalentClasses>
```

Con la representación realizada de los paquetes de red a través de la ontología, se obtiene una definición formal y estándar de Σ con la cual trabajaremos en las definiciones del resto de los elementos de *ModFunIDS*.

2.2 Tareas

Las tareas, es una de las piedras angulares de esta vista funcional del modelo. Desde el punto de vista formal serán definidas por la estructura mostrada en la ecuación 3, donde de cada una se describe el nombre, las precondiciones y las acciones que se llevaran a cabo si se cumplen las precondiciones [16].

$$t = \langle nombre, pre, acción \rangle \tag{3}$$

En esta:

nombre Expresión con la forma f(x1, ..., xk), donde cada xi es una variable autorizada para aparecer en las fórmulas pre y acción.

pre	Conjuntos de fórmulas condicionales que deberán ser evaluadas antes de ejecutar la acción.
acción	Conjuntos de fórmulas con la forma g(a1, ..., an), donde g es un predicado n-ario y cada ai son constantes o variables.

Por ejemplo, la tarea *Buscar PCs* puede describirse como se muestra a continuación:

nombre	Buscar PCs(ρ, M, MC, VP)
pre	Existe(ρ) && Preprocesado(ρ) && Entrenar(M)
acción	MatrizCovarianza(r, MC) VectPropios(MC, VP) ObtenerModelo(VP, M)

Donde: ρ es el conjunto de datos de entrenamiento, MC la matriz de covarianza, VP el conjunto de vectores propios de la matriz de covarianza y M el modelo PCA obtenido.

De esta manera las funciones de $Ejec_\alpha()$ de cada agente está definida por la ejecución de la tarea como se expresa a continuación:

$$Ejec=(\langle nombre, pre, acción\rangle, \phi_\alpha(t))=\begin{cases} acción \quad si\ pre((t))\ se\ verifica \\ \{\}\ si\ no\ se\ verifica \end{cases} \qquad 4$$

3 Agentes del IDS

Una vez definidos los aspectos concernientes al entorno de red es necesario definir los agentes que estarán regulando dicho entorno para mantenerlo libre de intrusiones. La definición de los agentes se realiza dejando muy claro los roles de cada actor, por lo que el *MAS* [17-19] que se describe posee una estructura predefinida y estática. De esta forma cada agente ejecutará un grupo de tareas concretas durante todo su ciclo de vida.

Un Sistema Multi-Agente es un sistema de alta complejidad resultando una tarea complicada su descripción. Por tanto, para lograr una mayor comprensión se ha dividido la explicación del *MAS*, correspondiente al *MIDS*, en diferentes modelos que permiten particionar la información del sistema. Para ello se parte de un Modelo Básico General que explica las características principales de la estructura y funcionamiento del *MAS*, luego se realiza una subdivisión, utilizando un criterio funcional, para explicar las principales características de los agentes del modelo.

3.1 Modelo básico general

Todos los agentes del *MAS* persiguen un objetivo común: detectar ataques de manera eficiente para mantener al entorno de red en un estado estable y libre de intrusiones. Si se define $_{Seg}\subseteq$ como un conjunto de estados del medio en el cual no existen ataques a la red, el objetivo general del *MAS* puede ser definido formalmente como: minimizar la distancia entre los estados del medio obtenidos a través de las influencias de los agentes y cualquier estado perteneciente al conjunto $_{Seg}$. Es decir, el objetivo del *MAS* es descrito como se muestra a continuación:

$$Min\ (Dist((t+1),_n))\ |\ _n \in {}_{Seg} \qquad 5$$

Independientemente de que tienen un objetivo común, los agentes según su tipo de funcionamiento se dividen en agentes operadores que son los encargados de llevar a cabo las tareas y agentes coordinadores que son aquellos que los supervisan. No obstante para este caso seremos más específicos con los agentes coordinadores, siendo importante el nivel de supervisión que poseen. Por tanto, definimos para el Sistema Multi-Agente una estructura organizativa jerárquica que cuenta con un Agente Coordinador General que es el encargado del funcionamiento de todo el sistema llevando a cabo los procesos da planificación y control. A éste se le subordinan los Coordinadores de Modelos que a su vez tendrán bajo su supervisión a otros Agentes Coordinadores de Operaciones que serán los que supervisen a los Operadores. Los Coordinadores de Modelo también pueden ser supervisores directos de algunos Agentes Operadores (figura 7).

El agente Coordinador General es el *agente IDS* (α_{IDS}), siendo el encargado de controlar todo el funcionamiento del sistema a través de su relación con los agentes supervisores de cada modelo: agente *Sensor*, coordinador del Modelo de Sensorización; agente *Reductor,* coordinador del Modelo de Reducción; agente *Clasificador*, coordinador del Modelo de Detección; agente *Administrador*; coordinador del Modelo de Respuesta; agente *Entrenador General,* coordinador del Modelo de Entrenamiento y el agente *Evaluador*, coordinador del Modelo de Autogestión. Adicionalmente existen relaciones directas entre los coordinadores de modelos para llevar a cabo su funcionamiento.

De esta forma el α_{IDS} responde al rol Coordinador General y es de tipo agente Interno. Las tareas que realiza para llevar a cabo su función principal, son de forma general la activación y desactivación de cada uno de los grandes procesos del modelo controlando el flujo global de trabajo, estas tareas son: *Activar sensorización, Activar reducción, Activar clasificación, Activar entrenamiento, Activar evaluación, Activar respuesta, Desactivar sensorización, Desactivar reducción, Desactivar clasificación, Desactivar entrenamiento, Desactivar evaluación* y *Desactivar respuesta*.

Así el agente α_{IDS} se relaciona de manera directa con los controladores de las principales funcionalidades, agrupadas por modelos, para garantizar su control como se muestra en el diagrama de interacción mostrado en la figura 8 utilizando un diagrama de colaboración AUML [20-22].

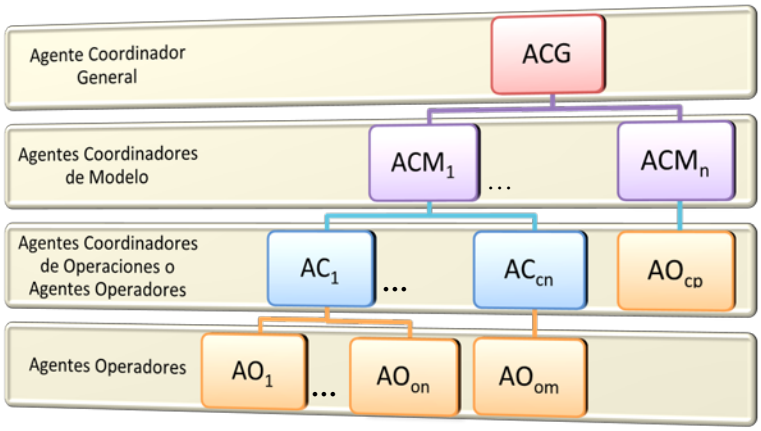

Fig. 7. Estructura Organizativa del MAS.

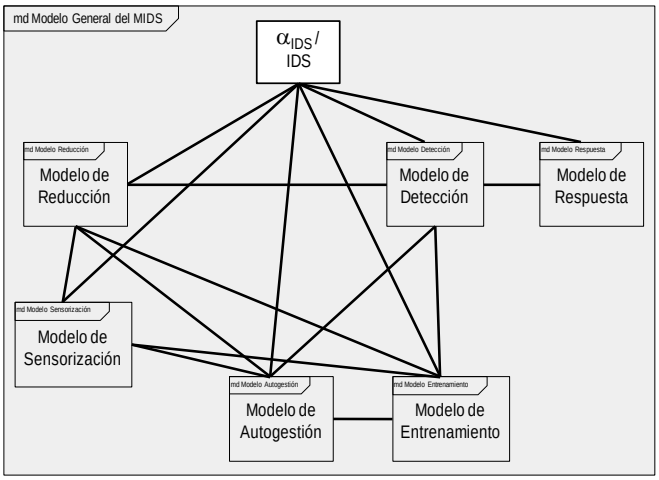

Fig. 8. Modelo Básico General del IDS.

Siguiendo el diagrama de la figura 8, se observa que el agente α_{IDS} emplea interfaces de salida para cada uno de los coordinadores de modelos como se muestra en la figura 9, donde se utiliza la notación BRIC.

A continuación se detallan el comportamiento y las características del resto de los agentes que han sido agrupados por funcionalidades.

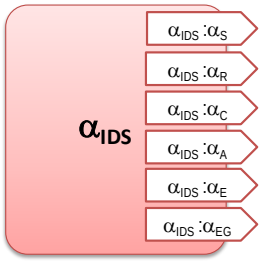

Fig. 9. Interfaces de comunicación del agente IDS.

3.2 Modelo de Sensorización

El *agente Sensor* ($_S$) es el encargado de controlar la captura y selección de datos de los paquetes TCP/IP que serán analizados. Para ello controla el funcionamiento del *agente Capturador* ($_{Cap}$) y el *Seleccionador de Datos* ($_{SD}$), lo que lo clasifica como un agente Interno bajo el rol de Coordinador de Modelo. De esta forma sus principales tareas son: *Iniciar captura, Iniciar selección de datos, Detener captura, Detener selección de datos, Enviar datos seleccionados al agente Reductor*.

El agente Capturador ($_{Cap}$) desempeña el rol de Operador y es el único agente de Interfaz de Entrada del modelo, sirviendo como punto de acceso de los datos del entorno de red hacia el resto de los agentes del *MIDS*. Las tareas fundamentales que desempeña son: Capturar tráfico y Enviar tráfico Capturado al Sensor. Por su relevancia, siguiendo la estructura PDE se define en la ecuación 6 la función de percepción de este agente que es el encargado de capturar los paquetes del tráfico TCP/IP que circula por la red.

$$Percept_{Cap}(\sigma_i) = {}_{cap_i} \quad donde \quad {}_{cap_i} = \sigma_i.paqIP \mid \sigma_i \in \qquad 6$$

De esta manera a $_{Cap}$ llegan todos los paquetes definidos dentro del estado del mundo y éste percibe sólo aquéllos que son de interés para el IDS, paquetes IP, siguiendo la definición realizada en la ontología que describe los estados del mundo. Luego, la deliberación del agente se puede expresar formalmente como:

$$Delib_{Cap}({}_{Cap_i}, s_{Cap_i}) = t_i \mid Delib_{Cap} \begin{cases} t_i = CapturarTráfico \\ si\ CapturaActiva \\ \text{or} \\ t_i = EnviarTrafCapSensor({}_{Cap_i}) \\ si\ Correct({}_{Cap_i}) \end{cases} \qquad 7$$

Por su parte, el agente Seleccionador de Datos ($_{SD}$) recibe los datos de los paquetes TCP/IP capturados. Si los datos llegan correctos se aplican las reglas de selección sobre los paquetes para obtener los paquetes seleccionados. Los paquetes seleccionados no contienen todos los datos de las cabeceras TCP/IP sino que incluyen solamente los datos definidos en las reglas. Por tanto, las reglas de selección forman

parte de la memoria interna (S_{SD}) de este tipo de agente. De forma general, se puede decir que las tareas principales de este agente son: Seleccionar datos y Enviar datos seleccionados al Sensor.

De esta forma el diagrama de interacción que describe las relaciones de estos agentes es el que se muestra en la figura 10.

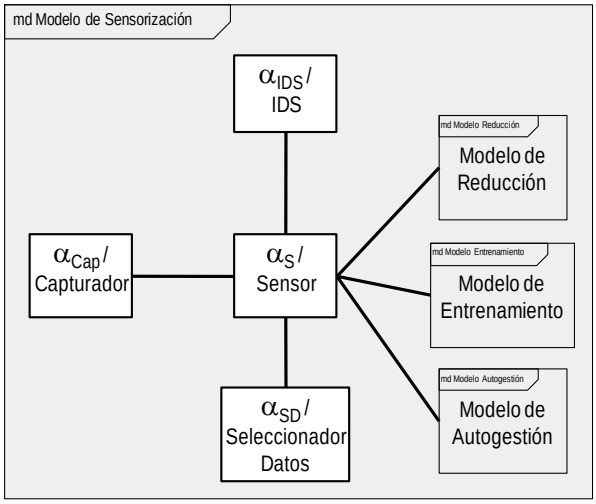

Fig. 10. Diagrama de interacción del Modelo de Sensorización.

Las relaciones descritas en la figura 10 se corresponden con las interfaces definidas para cada uno de los agentes involucrados en la captura (figura 11). Adicionalmente el agente Sensor tiene interfaces de entrada para los agentes IDS, Reductor, Entrenador y Evaluador como parte de las relaciones entre modelos.

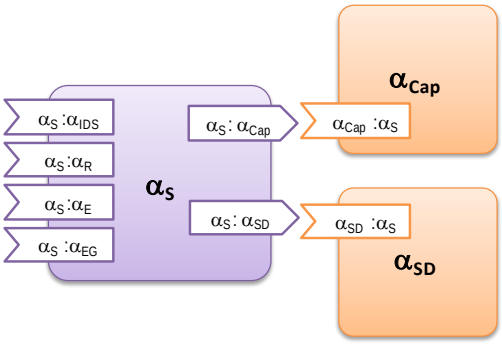

Fig. 11. Interfaces de comunicación de los agentes del Modelo de Sensorización.

3.3 Modelo de Reducción

Este modelo es sumamente importante dentro del *MIDS* ya que aporta la eficiencia del sistema llevando a cabo la reducción de características de entrada al clasificador a través de la reducción de los paquetes de red.

El *Reductor* ($_R$) recibe del $_S$ los paquetes capturados y los envía al $_{PCA}$ para que éste los reduzca. Luego $_{PCA}$ utiliza el *Modelo PCA*, que tiene almacenado como parte de su memoria interna (S_{PCA}), transformando los datos de los paquetes seleccionados con las nuevas bases definidas por el *Modelo PCA*. La función de deliberación de este agente se define formalmente en la ecuación 8 donde se muestran además las tareas realizadas por el mismo:

$$Delib_{PCA} = \begin{cases} t_i = FiltrarPCA(_{PCA_i}.paqIP, \ _r) \\ si\ Etiq(_{PCA_i}) = paqIP\ \&\&\ Correct(s_{PCA_i}.ModPCA) \\ or \\ t_i = EnviarDatosReducidos(_r) \\ si\ Correct(_r) \\ or \\ t_i = CambiarModelo(_{PCA_i}.ModPCA) \\ si\ Etiq(_{AC_i}) = ModPCA\ \&\&\ Correct((_{PCA_i}.ModPCA) \end{cases} \quad 8$$

donde δ_r son los datos reducidos.

La última parte de la función de deliberación 8 será explicada más adelante junto con los agentes de Entrenamiento.

Es el agente *Reductor* ($_R$) el encargado de sincronizar entre sí a los agentes internos de su modelo además de sincronizarlos con los modelos externos de Entrenamiento y Autogestión, para garantizar que el *Modelo PCA* esté entrenado y actualizado con respecto al entorno de red. Luego, este agente coordinador de modelo tiene como principales tareas: Iniciar reducción, Solicitar cambio de modelo PCA, Solicitar modelo PCA, Enviar datos reducidos al *agente Clasificador.*

El diagrama de interacción AUML de la figura 12 muestra las relaciones de los agentes de este modelo y en la figura 13 se muestran las interfaces tanto de entrada como de salida que implementan cada uno de estos agentes para utilizar los canales de comunicación necesarios.

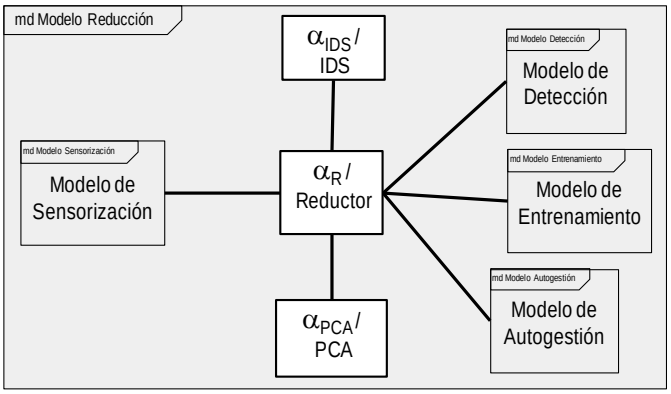

Fig. 12. Diagrama de interacción del Modelo de Reducción.

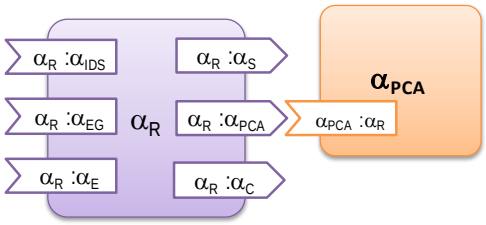

Fig. 13. Interfaces de comunicación de los agentes del Modelo de Reducción.

3.4 Modelo de Detección

Este modelo es sumamente importante dentro del *MIDS* ya que aporta la eficiencia del sistema llevando a cabo la reducción de características de entrada al clasificador a través de la reducción de los paquetes de red.

En este modelo se agrupan los agentes encargados de la clasificación de los paquetes que circulan por la red.

El agente Clasificador ($_C$) recibe del agente Reductor los datos a clasificar (δr) y los entrega al agente SOM ($_{SOM}$) que utiliza los datos de su memoria interna (SSOM) para utilizar el Modelo SOM y clasificar los paquetes. Luego, la función de deliberación de $_{SOM}$ es muy similar a la de $_{PCA}$ como se muestra en 9 donde se muestran también las tareas relacionadas con este agente.

$$Delib_{SOM} = \begin{cases} t_i = ClasificarTráfico(_{SOM_i}.\delta_r) \\ si\ Etiq(_{SOM_i}) = paqIP\ \&\&\ Correct(s_{SOM_i}.ModSOM) \\ or \\ t_i = CambiarModelo(_{SOM_i}.ModSOM) \\ si\ Etiq(_{AC_i}) = ModSOM\ \&\&\ Correct((_{SOM_i}.ModSOM) \end{cases} \quad 9$$

La segunda parte de la función también será explicada en el modelo de entrenamiento.

La función de ejecución definida para cada agente es la descrita en 5.4, por lo que para el agente SOM basta la definición siguiente de Clasificar tráfico para entender el funcionamiento de la ejecución de dicha tarea.

nombre *Clasificar tráfico(δr, MSOM, VE, VEp, ID)*

pre *Existe(δr) && Entrenado(MSOM)*

acción *Vect Entrada(δr, VE)*
Preprocesar(VE, VE_p)
EjecutarSOM(VE_p, MSOM, ID)

Donde: δr son los datos reducidos, MSOM el modelo SOM que contiene la arquitectura de la ANN y su distribución de pesos, VE el vector de entrada, VE_p el vector de entrada preprocesado e ID los informes de detección.

Por tanto, el Coordinador de Modelo, agente *Clasificador*, es el responsable del correcto funcionamiento de la SOM, así como de su sincronización con el resto de los modelos llevando a cabo las siguientes tareas: *Iniciar clasificación, Solicitar cambio de modelo SOM, Solicitar modelo SOM y Enviar informes de detección al agente Administrador.*

El diagrama de colaboración AUML que describe la interacción entre los agentes de este modelo se muestra en la figura 14.

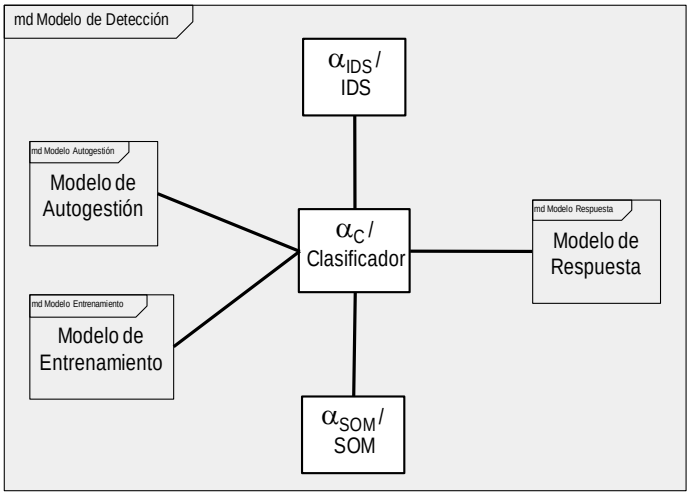

Fig. 14. Diagrama de interacción del Modelo de Detección.

Las interfaces que implementan cada agente para establecer los canales de comunicación necesarios para mantener las relaciones mostradas en la figura 14, se representan en la figura 15.

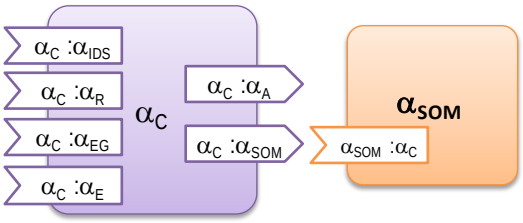

Fig. 15. Interfaces de comunicación de los agentes del Modelo de Clasificación.

3.5 Modelo de Respuesta

Este ha sido el modelo menos desarrollado dentro de la investigación, ya que el *MIDS* se centra en los mecanismos necesarios para la detección eficiente de ataques. A este nivel se asume que todo el Modelo de Respuesta puede ser sustituido por un agente humano (administrador de red) que actúe sobre la red tras el análisis de los Informes de detección.

Dentro del modelo está concebido que el agente *Clasificador* entregue al *Administrador* ($_{A}$) los Informes de detección para hacerlos llegar al *Analizador* ($_{An}$). Este último analiza los informes y aquellos paquetes que hayan sido clasificados como intrusivos se los envía al *Ejecutor* ($_{Ej}$). En la figura 16 se muestra un diagrama de colaboración que describe la interacción entre éstos agentes.

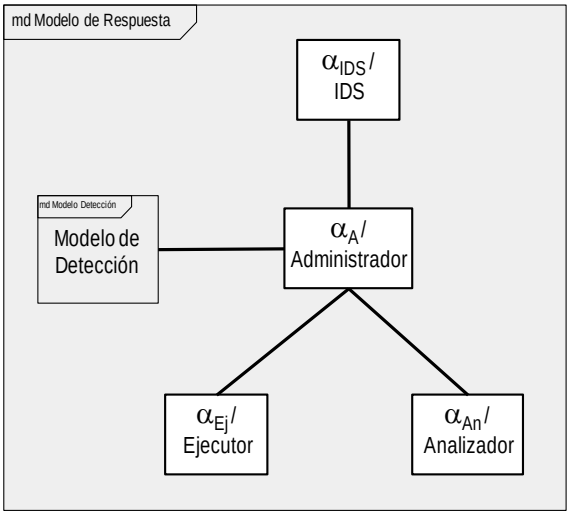

Fig. 16. Diagrama de interacción del Modelo de Respuesta.

De esta forma las tareas llevadas a cabo por el *Administrador* para garantizar el flujo de eventos son: *Iniciar análisis de informes y Enviar intrusiones detectadas al Ejecutor*. Por su parte, el agente *Analizador* lleva a cabo las tareas: *Analizar informes de detección y Enviar intrusiones al Administrador.* Por último el Ejecutor reacciona ante las intrusiones a través de la tarea *Responder* y utilizando las *Reglas de Respuesta* que tiene en su memoria interna (S_{Ej}) para actuar sobre la red según sea necesario en cada caso. Este agente es del tipo Interfaz de Salida ya que es el que actúa directamente sobre el entorno de red.

Las relaciones establecidas entre los agentes son llevadas a cabo a través de las interfaces de comunicación que cada uno de ellos posee como se muestra en la figura 17.

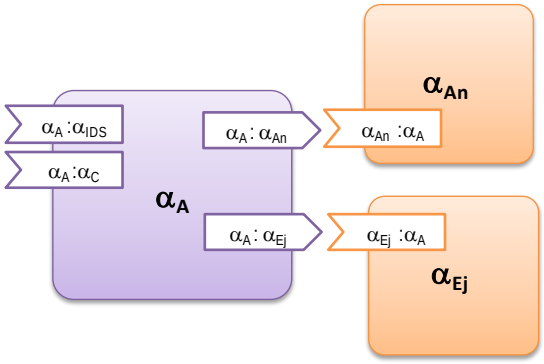

Fig. 17. Interfaces de comunicación de los agentes del Modelo de Respuesta.

3.6 Modelo de Entrenamiento

Tanto el Modelo de Reducción como el de Detección utilizan técnicas que precisan de entrenamiento previo. Siguiendo las metas trazadas para el diseño del *MIDS*, éste debe ser de mantenimiento mínimo, por tanto es importante que contemple un conjunto de agentes que sean los encargados de entrenar los algoritmos de manera no supervisada.

El Modelo de Entrenamiento es el responsable de entrenar tanto al PCA como a la SOM. Este es el primer modelo que es activado por el *Coordinador General IDS* para garantizar que el *Reductor* y el *Clasificador* comiencen a funcionar con los algoritmos para PCA y SOM correctamente entrenados.

El *Entrenador General* ($_{EG}$) se comunica con el *Reductor* y el *Clasificador* para enviarles los Modelos PCA y SOM que necesitan respectivamente para su funcionamiento. Por esta razón el agente *Reductor* enviará al agente *PCA* un *Modelo PCA* para que actualice su memoria interna, de esta forma se explica totalmente la función de deliberación definida en 8. De forma análoga, el *Clasificador* puede solicitar el cambio del *Modelo SOM* del agente *SOM*, lo que explica también la última parte de la ecuación 9.

El $_{EG}$ se comunica con el *Sensor* para recibir de éste los paquetes necesarios que conforman los conjuntos de entrenamiento y entregarlos tanto al *Entrenador PCA* ($_{EPCA}$) como al *Entrenador SOM* ($_{ESOM}$). En el caso de éste último el conjunto de

entrenamiento precisa ser reducido, por lo que el $_{EG}$ tiene que solicitar al *Reductor* que reduzca el conjunto de entrenamiento SOM para que pueda ser utilizado por el agente *Buscador Modelo SOM* ($_{BSOM}$).

Para llevar a cabo el rol de Coordinador de Modelo que desempeña, el $_{EG}$ es el responsable de tareas como: *Iniciar entrenamiento PCA, Iniciar Entrenamiento SOM, Solicitar paquetes al Sensor, Solicitar reducción de paquetes al Reductor.* De esta forma las interfaces de comunicación de $_{EG}$ son las que se muestran en la figura 18.

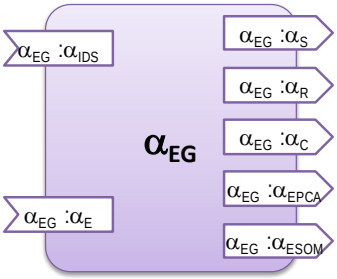

Fig. 18. Interfaces de comunicación del agente Entrenador General.

El $_{EPCA}$, una vez activado, solicita al *Seleccionador de paquetes* ($_{SP}$) el conjunto de paquetes necesarios para el entrenamiento del Modelo PCA. El $_{SP}$ utiliza su memoria interna (S_{SP}) para obtener el algoritmo de selección (*Muestreo*) que debe implementar y solicita al $_{EPCA}$ un conjunto de paquetes capturados. Una vez generado el Conjunto de paquetes se los trasmite al *Preprocesador PCA* que pone a punto los datos para ser utilizados por el agente *Buscador PCs* ($_{BPC}$) llevando a cabo las tareas: *Preprocesar datos y Enviar datos preprocesados al EntrenadorPCA*. El $_{BPC}$ genera y ejecuta la tarea *Buscar PCs* y la tarea *Enviar el Modelo PCA obtenido al Entrenador PCA*. Así, el $_{EPCA}$, tiene como tareas: *Solicitar conjunto paquetes al Entrenador General, Enviar conjunto de entrenamiento al Preprocesador, Enviar datos procesados al Buscador PCs y Enviar Modelo PCA al Entrenador General*. Las relaciones necesarias para llevar a cabo las tareas de entrenamiento del Modelo PCA establecen las interfaces definidas en la figura 19.

De forma similar, el *Entrenador SOM* supervisa la generación de conjuntos de entrenamiento, preprocesamiento y obtención de su modelo a través de la ejecución de las tareas: *Solicitar conjunto paquetes al Entrenador General, Solicitar reducción de paquetes al Entrenador General, Enviar conjunto de entrenamiento al Preprocesador SOM, Enviar datos procesados al Buscador SOM y Enviar Modelo SOM al Entrenador General.* Los agentes operadores bajo su coordinación funcionan de manera similar a los encargados del entrenamiento de PCA. El Generador Conjunto ($_{GC}$) *Solicita paquetes SOM al Entrenador SOM* y luego de seleccionados *Solicita la reducción de los datos al Entrenado SOM* y por último *Envía el conjunto de entrenamiento SOM al Entrenador SOM*. Por su parte el *Preprocesador SOM* ($_{PSOM}$) preprocesa los datos y los envía al ($_{ESOM}$) y el *Buscador Modelo SOM* responde a las tareas: *Buscar modelo SOM y Enviar Modelo SOM al EntrenadorSOM.* Las interfaces correspondientes a los canales de comunicación descritos se muestran en la figura 20.

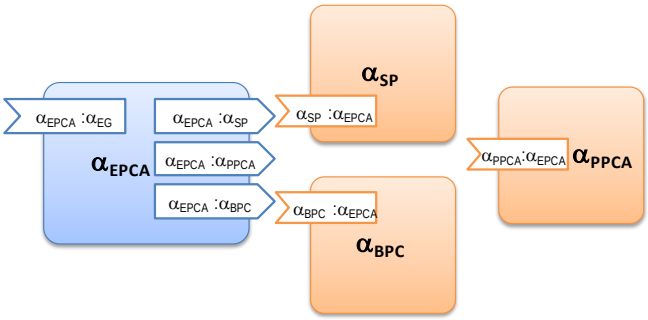

Fig. 19. Interfaces de comunicación de los agentes de entrenamiento PCA.

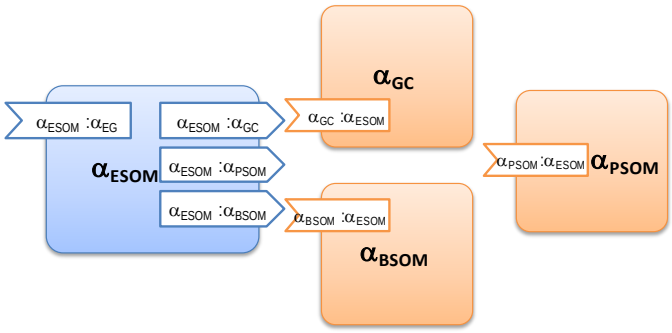

Fig. 20. Interfaces de comunicación de los agentes de entrenamiento SOM**.**

La figura 21 muestra una vista global de las interacciones entre los agentes de este modelo de entrenamiento a partir del diagrama de colaboración AUML que le corresponde.

3.7 Modelo de Autogestión

Este modelo contiene agentes que ofrecen el dinamismo necesario para lograr que los algoritmos de reducción y detección funcionen con modelos actualizados.

El Modelo de Autogestión tiene características diferentes al resto de los modelos. Esto se debe a que su funcionalidad debería estar contenida en cada uno de los modelos anteriores, pero por la importancia del dinamismo, siendo uno de los objetivos estratégicos del *MIDS*, lo planteamos como un modelo independiente que controla la autogestión de los coordinadores del resto de los modelos. Las relaciones principales y los agentes implicados se muestran en la figura 22.

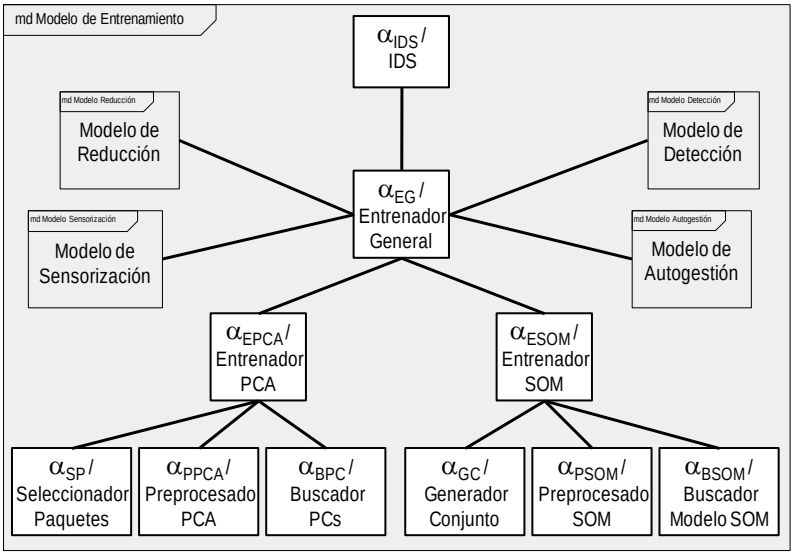

Fig. 21. Diagrama de interacción del Modelo de Entrenamiento.

El Coordinador de este modelo es el agente *Evaluador* ($_E$), que es el encargado de obtener los modelos de reducción y detección, mandarlos a evaluar y en caso de que sea necesario re-entrenarlos y sustituirlos. De esta forma $_E$ obtiene el Modelo PCA a través del *Reductor* y lo envía al *Evaluador de Modelo* ($_{EM}$), que lo evalúa según lo definido en al capítulo anterior, comparando la desviación del Modelo PCA respecto a un conjunto de paquetes actuales y calculando la diferencia entre dos conjuntos de paquetes tomados en diferentes ventanas de tiempo. Luego se evalúa si existe un cambio en el comportamiento de la red que amerite re-entrenar el Modelo de Reducción y el Modelo de Clasificación. De esta manera la función de deliberación del agente $_{EM}$ queda definida en la ecuación 10 así como sus principales tareas.

$$Delib_{EM} = \begin{cases} t_i = Solicitar\ conjunto\ paq\ al\ Evaluador() \\ si\ IniciarEvaluación \\ and \\ t_i = Calcular\ Dif() \\ si\ Corret(_i) \\ and \\ t_i = Calcular\ Dev() \\ si\ (Dif(_{EM_i} \cdot 1, _{EM_i} \cdot 2) > _1 \\ and \\ t_i = Solicitar\ RecalculoModelo() \\ si\ (Dif(_{EM_i} \cdot 1, _{EM_i} \cdot 2) > _1\ or\ Dev(_{EM_i} \cdot 1, M) > _2) \end{cases} \quad 10$$

Donde ρ_i son los conjuntos de paquetes y *M* el Modelo PCA.

Para ello $_{E}$ debe enviar al $_{EM}$ tanto el modelo proveniente del *Reductor* como los conjuntos de paquetes necesarios para la evaluación provenientes del *Modelo de Entrenamiento*. Una vez evaluado si es necesario re-entrenar los modelos se generan tareas dirigidas el *Entrenador General* para que entrene un nuevo modelo PCA y un nuevo Modelo SOM y los envíe a los Modelos de Reducción y Clasificación respectivamente. Por tanto las tareas fundamentales del *agente Evaluador* son: *Solicitar Modelo PCA al Reductor, Solicitar paquetes al Sensor, Enviar Modelo PCA al EvaluadorModelo, Enviar paquetes al EvaluadorModelo, Generar Tareas, Enviar tarea al EntrenadorGeneral.*

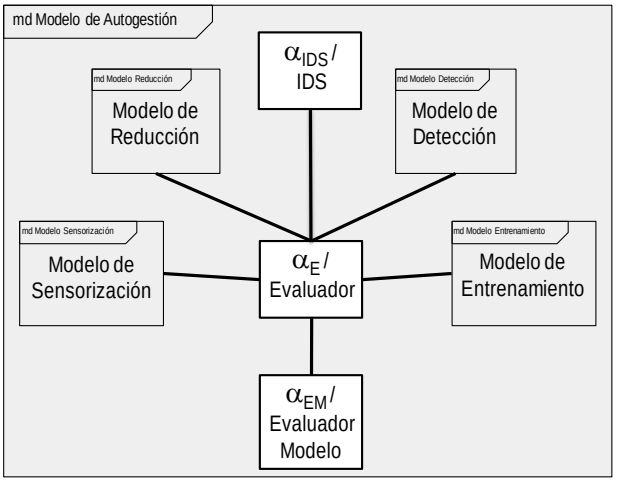

Fig. 22. Diagrama de interacción del Modelo de Autogestión.

Para llevar a cabo sus funcionalidades y mantener la comunicación entre los agentes implicados dentro del modelo, las interfaces empleadas por los agentes evaluadores se muestran en la figura 23.

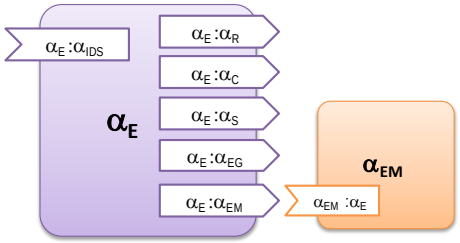

Fig. 23. Interfaces de comunicación de los agentes del Modelo de Evaluación.

4 Conclusiones

A lo largo del artículo se han explicado cada uno de los elementos involucrados en la definición funcional del *MIDS* a través del modelo de acción-reacción.
Los estados del mundo fueron definidos de manera formal a través de una ontología que permitió describir los paquetes de red y sus relaciones. Por otra parte se definieron de manera estructurada las tareas que pueden ser ejecutadas sobre el medio, para lograr un carácter algorítmico del Modelo.

Los agentes reguladores del entorno de red se representaron agrupados por modelos que engloban las principales funcionalidades del sistema. Logrando de esta forma una descripción general y detallada del funcionamiento del *MIDS*.

De esta forma se logró modelar un Sistema de Detección de Intrusos, que aplica PCA como técnica de reducción de características y SOM como clasificador, de forma distribuida basándose en la teoría de Sistemas Multiagentes para obtener un algoritmo computable del Modelo.

Referencias

1. Computer Word, *Las TIC se han vuelto imprescindibles*, No 1212, pag 16, España, abril 2009.
2. Iws. *Internet World Stats*. http://www.internetworldstats.com/stats.htm, 2008.
3. IBM. *Internet Security Systems*, http://www.iss.net/, 2008
4. Mchugh, J., Christie, A., Allen, J.: Defending yourself: the Role of Intrusion Detection Systems. IEEE Software. 17, 42--51 (2000)
5. Kruegel, Ch., Valeur, F. & Vigna, G. (2005). "Instrusion Detection and Correlation. Challenges and Solutions". ISBN 0-387-23398-9.
6. Dacier, M., Debar, H. &Wespi, A. (1999). "Towards taxonomy of intrusion-detection systems". Computer Networks, 31 (8):805–822.
7. INE. (2009). Instituto Nacional de Estadística, http://www.ine.es. [noviembre 2009].
8. Lorenzo, I., Maciá, F., Mora, F.J., Marcos, D., Gil, J.A. & Lau, R., (2009). "Marco Formal para el Modelado de un Sistema de Detección de Intrusos de Red." *In proc of Desarrollo de Grandes Aplicaciones de Red. VI Jornadas, JDARE 2009. Alicante, España, octubre 15–16, 2009. Actas.*
9. López, J., Villagrá, V. & Berrocal, J. (2004). "Applying the WebOntology Language to management information definitions." *In IEEE Communications Magazine*, vol. 42, no. 7, pp. 68–74.
10. López, J., Villagrá, V. & Berrocal, J. (2005). "Application of OWL-S to define management interfaces based on Web Services." *In proc of 8th IFPI/IEEE International Conference on Management of Multimedia Networks and Services*, LNCS vol.3754, pp. 24–26.
11. Klie, T., Gebhard, F. & Fischer, S. (2007). "Towards automatic composition of Network Management Web Services." *In proc of International Symposium on Integrated Network Management, IM'07, 10th IFPI/IEEE. Munich.*
12. Hiu, X. & Debao, X (2006). "A common Ontology- based Intelligent Configuration Management Model for IP-Network Devices." *In proc. of International Conference on Innovative Computing, Information and Control'2006.*
13. Strassner, J., Neuman, J., Van der Meer,S., Davy, S., Barret, K., Raymer, D. & Samudrala, S. (2009) "The Design of a New Policy Model to Support

Ontology-Driven Reasoning for Autonomic Networking." *In Journal of Network and Systems Management*, Springer.
14. Protégé (2006). Editor de Ontologías y Sistema de Adquisición de Conocimiento, http://protege.standford.edu/
15. Horridge, M. (2009). "A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools".Edition 1.2, University Of Manchester.
16. Maciá, F. & García, J. (2006) "Mobile Agent System Framework Suitable for Scalable Networks." *Kybernetes. The International Journal of Systems and Cybernetics*, vol. 35, no. 5, pp. 688–699.
17. Cabac, L. & Moldt, D. (2004)." Formal Semantics forAUML Agent Interaction Protocol Diagrams".Department of Computer Science, TGI, University of Hamburg
18. A Cabac, L.(2003)." Modeling Agent Interaction with AUML Diagrams and Petri Nets". Diplomarbeit, University of Hamburg, Department of Computer Science, Vogt-KÄolln Str. 30, 22527 Hamburg, Germany
19. Ferber, J. (1999). "Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence." *Addison-Wesley*, ISBN 0-201-36048-9.
20. Labrou, Y. & Finin, T. (1998)."Semantics and Conversions for an Agent Communication Language". In Readings in Agents, Huhns, M y Singh, M. (Eds.) Morgan Kaufmann, pp. 235-242.
21. Smith, R.G. (1980). " The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver". IEEE Trans. On Computers. Vol. 29, cap. 12, pp. 1104-1113.
22. Davis, R. &Smith, R.J. (1983). "Negotiation as a Metaphor for Distributed Problem-Solving". Artificial Intelligence. 20(1), pp. 63-109.